CLAIM AMENDMENTS

1. (original) A method for repetitively executing a plurality of software packages at one or more rates, utilizing a common set of computational resources, the method comprising the steps:

generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages;

A
)

executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals.

2. (original) The method of claim 1 wherein the plurality of software packages of the "executing" step includes only valid software packages, the method further comprising the step:

utilizing one or more tests to identify the software packages that are valid.

3. (original) The method of claim 2 wherein one of the tests for validity is a one's complement checksum test of a software package's program memory.

4. (original) The method of claim 2 wherein a software package is assigned its own dedicated memory region, one of the tests for validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region.

5. (original) The method of claim 4 wherein one of the tests is whether the address is returned within a predetermined time.

6. (original) The method of claim 2 wherein a software package is assigned its own dedicated memory region, the software package's dedicated memory region including a stack memory region and/or a heap memory region, one of the tests for validity being whether the stack memory range and/or the heap memory range assigned during the execution of the software

package's initialization procedure and the various associated entry points lies within the software package's dedicated memory region.

7. (original) The method of claim 6 wherein one of the tests is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time.

8. (original) The method of claim 1 wherein a software package is assigned its own dedicated memory region.

9. (original) The method of claim 8 wherein the software package's dedicated memory region includes a stack memory region, a software package's stack residing in the software package's stack memory region.

10. (original) The method of claim 1 wherein a software package includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed.

11. (original) The method of claim 10 wherein a background task is an infinite loop.

12. (original) The method of claim 10 wherein the software package causes the power utilized in executing the software package to be minimized after completion of the background tasks.

13. (original) The method of claim 1 wherein a failure in the execution of a software package causes information to be logged in a failure log.

14. (original) The method of claim 13 wherein a failure in execution is linked to the software package that caused the failure.

15. (original) The method of claim 13 wherein quality of performance in executing a software package is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in a

failure log, the execution of a software package being subject to a plurality of execution options, an execution option being selected on the basis of one or more performance-quality parameter values.

16. (original) The method of claim 15 wherein the plurality of execution options are user configurable.

17. (original) The method of claim 15 wherein performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in a software package's failure log.

18. (original) The method of claim 1 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software.

19. (currently amended) The method of claim 1 wherein each of the plurality of software packages is assigned its own memory block, a software package being ~~enabled~~ enableable to read data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks readable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.

20. (currently amended) The method of claim 1 wherein each of the plurality of software packages is assigned its own memory block, a software package being ~~enabled~~ enableable to write data only to zero or more memory blocks associated with other software packages, the zero or more memory blocks writeable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.

21. (original) The method of claim 1 wherein an executive software package enforces the discipline that each software package executes only during the time intervals of its sequence of time intervals, the executive software package determining when the execution of a software package extends into a time interval belonging to the sequence of time intervals assigned to another software package and performs a remedial action.

22. (original) The method of claim 1 wherein the presence of those software packages that are present is detected.

23. (original) The method of claim 1 wherein one or more of the plurality of software packages are independently compiled, linked, and loaded.

24. (original) The method of claim 1 wherein a software package has its own stack, the software package's stack being selected prior to executing the software package.

25. (original) Apparatus for practicing the method of claim 1.

26. (original) Apparatus for repetitively executing a plurality of software packages at a plurality of rates, the apparatus comprising:

a means for generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages;

a means for executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals.

27. (original) The apparatus of claim 26 wherein the plurality of software packages executed by the "executing" means includes only valid software packages, the apparatus further comprising:

a means for utilizing one or more tests to identify the software packages that are valid.

28. (original) The apparatus of claim 27 wherein one of the tests for validity is a one's complement checksum test of a software package's program memory.

29. (original) The apparatus of claim 27 wherein a software package is assigned its own dedicated memory region, one of the tests for validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region.

30. (original) The apparatus of claim 29 wherein one of the tests is whether the address is returned within a predetermined time.

31. (original) The apparatus of claim 27 wherein a software package is assigned its own dedicated memory region, the software package's dedicated memory region including a stack memory region and/or a heap memory region, one of the tests for validity being whether the stack memory range and/or the heap memory range assigned during the execution of the software package's initialization procedure and the various associated entry points lies within the software package's dedicated memory region.

32. (original) The apparatus of claim 31 wherein one of the tests is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time.

33. (original) The apparatus of claim 26 wherein a software package is assigned its own dedicated memory region.

34. (original) The apparatus of claim 33 wherein the software package's dedicated memory region includes a stack memory region, a software package's stack residing in the software package's stack memory region.

35. (original) The apparatus of claim 26 wherein a software package includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed.

36. (original) The apparatus of claim 35 wherein a background task is an infinite loop.

37. (original) The apparatus of claim 35 wherein the software package causes the power utilized in executing the software package to be minimized after completion of the background tasks.

38. (original) The apparatus of claim 26 wherein a failure in the execution of a software package causes information to be logged in a failure log.

39. (original) The apparatus of claim 38 wherein a failure in execution is linked to the software package that caused the failure.

40. (original) The apparatus of claim 38 wherein quality of performance in executing a software package is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in a failure log, the execution of a software package being subject to a plurality of execution options, an execution option being selected on the basis of one or more performance-quality parameter values.

41. (original) The apparatus of claim 40 wherein the plurality of execution options are user configurable.

42. (original) The apparatus of claim 40 wherein performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in a software package's failure log.

43. (original) The apparatus of claim 26 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software.

44. (currently amended) The apparatus of claim 26 wherein each of the plurality of software packages is assigned its own memory block, a software package being ~~enabled~~ enableable to read data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks readable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.

45. (currently amended) The apparatus of claim 26 wherein each of the plurality of software packages is assigned its own memory block, a software package being ~~enabled~~ enableable to write data only to zero or more memory blocks associated with other software packages, the zero or more memory blocks writeable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.

46. (original) The apparatus of claim 26 wherein an executive software package enforces the discipline that each software package executes only during the time intervals of its sequence of time intervals, the executive software package determining when the execution of a software package extends into a time interval belonging to the sequence of time intervals assigned to another software package and performs a remedial action.

47. (original) The apparatus of claim 26 wherein the presence of those software packages that are present is detected.

48. (original) The apparatus of claim 26 wherein one or more of the plurality of software packages are independently compiled, linked, and loaded.

49. (original) The apparatus of claim 26 wherein a software package has its own stack, the software package's stack being selected prior to executing the software package.